Dynamic Theoretical Analysis of the Distributed Stochastic and Distributed Breakout Algorithms

Anton Ridgway
University of Tulsa
Tandy School of Computer Science
Tulsa, Oklahoma
anton-ridgway@utulsa.edu

Roger Mailler
University of Tulsa
Tandy School of Computer Science
Tulsa, Oklahoma
roger-mailler@utulsa.edu

ABSTRACT

The distributed constraint satisfaction problem is often used to model real-world situations and find agent-based solutions. A number of methods have been developed to solve these problems, including the well-known DSA and DBA algorithms. In many real scenarios, however, the problems are not static. This forces practitioners to adapt these protocols to solve dynamic distributed constraint satisfaction problems (DynDCSP). Surprisingly, despite long-running study of the problem, practically all analysis of DynDCSP algorithms has been experimental in nature. This work presents a new theoretical assessment of the DSA and DBA algorithms, leveraging a mapping of DynDCSP instances to a physical thermodynamics model to develop a deeper understanding of the algorithms' behavior.

Here, we assess the static versions of DSA and DBA, but focus on examining their rates of convergence, not just their final convergence points, as a means of understanding how they will perform in dynamic settings. We develop various theoretical approaches to show how the algorithms' convergence rates are affected by problem density and tightness, and examine the impact that problem size has on an algorithm's performance. Finally, we show the accuracy of our analytical predictions through comparison with experimental results.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence-Multiagent systems, Coherence and coordination

General Terms

Algorithms, Performance, Experimentation, Theory

Keywords

Dynamics, Constraint Satisfaction, Thermodynamics

1. INTRODUCTION

In studying the applications of distributed constraint satisfaction problems (DCSP), it is often crucial to have algorithms available to us that are not only able to solve problems, but also able to adapt to a changing problem space. This has naturally led to the expansion of DCSP studies to the realm of dynamic distributed constraint satisfaction problems (DynDCSP). Under the DynDCSP model described by Mailler [6], these problems can be defined as DCSPs whose constraints change continuously over time during the problem solving process. This definition differs from the original definition presented by Dechter and Dechter [3], where changes to the problem occurred between subsequent executions of a problem solver. Several algorithms have been developed or modified to solve this problem, including the Distributed Breakout Algorithm (DBA) [11, 6], the Distributed Stochastic Algorithm [5] and the Asynchronous Partial Overlay (APO) protocol [6], but in every case the complex nature of the problem has limited the evaluation of these algorithms to empirical analysis.

In a recent work by Mailler and Zheng [7], they showed that DynDCSPs can be mapped to physical systems and therefore obey the laws of thermodynamic theory. This is an important finding because it allows researchers and practitioners to characterize an environment and protocol based on the rate at which they affect a problem and predict the equilibrium that the combination would establish. Their findings considerably reduce the amount of empirical evaluation needed to compare dynamic protocols, but still rely on measuring their behavior on static instances.

Moreover, this analysis drew attention to another important property of DynDCSP algorithms, which had not been considered important in the past. Not only should these algorithms be considered in terms of the quality of the solution they converge on (convergence point), but also in terms of how fast they reach these values (convergence rate). Consideration of both the rate and the point are crucial in determining which algorithms are best suited for any given application.

In this paper, we perform a theoretical analysis of DSA and DBA, two simple and well-known algorithms whose behavior is nevertheless challenging to analyze. Our analysis will focus on the impact that a problem's density, tightness, and size have on the convergence rate of these protocols. The results of this are bounds and approximations on these protocols' performance in dynamic environments. These bounds are then compared with empirical results to validate their accuracy.

The rest of this paper is organized as follows: In section 2, we will discuss background including introducing the DynDCSP problem, some useful terminology, and the mapping of DynDCSPs to thermodynamic systems. In section 3, we will reacquaint the reader with the DSA and DBA protocols and provide our analysis of their performance. This section also includes our empirical results including the test setup, findings, and comparison to our theoretical bounds. Finally, in section 4, we will present concluding remarks and provide some areas for future research.

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

2. BACKGROUND AND TERMINOLOGY

In general, a static DCSP is described by $P =< V, A, D, C >$, where the following is true:

- V is a set of n variables: $V = \{v_1, \ldots, v_n\}$.
- A is a set of g agents: $A = \{a_1, \ldots, a_g\}$.
- D is a set of discrete, finite domains for each variable: $D = \{D_1, \ldots, D_n\}$.
- C is a set of constraints $C = \{c_1, \ldots, c_m\}$, where each $c_i(d_{i,1}, \ldots, d_{i,j})$ is a function $c_i : D_{i,1} \times \cdots \times D_{i,j} \rightarrow \{true, false\}$. That is, a constraint returns true if the values for its associated variables are such that the constraint is satisfied.

Given this, we seek to find a solution $\mathcal{S}^* = \{d_1, \ldots, d_n | d_i \in D_i\}$ such that no constraint is violated, or otherwise, to identify that this cannot be done. Each agent a_i is associated with at least one variable and its variables' constraints, and is primarily concerned with finding a violation-free solution for its own portion of the problem space. Agents coordinate together to solve their constraints, leading to a global problem solution.

Note that in this paper, unless otherwise stated, we consider binary constraints only. In general, it is straightforward to generalize n-ary constraints to binary [1], and while this does affect how an algorithm approaches a given n-ary constraint problem, it will still perform within the bounds we develop here.

We also make use of the following important variable assignments, commonly seen in the literature [8]:

- p_1 is the density of the problem. This indicates the percent of possible constraints that are actually present. Given n variables, we know that $\frac{n(n-1)}{2}$ constraints are possible, therefore $m = p_1 \frac{n(n-1)}{2}$.
- p_2 is the average tightness of the variables in the problem. This term represents the difficulty of an individual constraint being satisfiable, in terms of the percentage of constraint configurations that evaluate to $false$.
- The expected number of initially-violated constraints is $mp_2 = p_1 p_2 \frac{n(n-1)}{2}$.
- The degree of a problem is the average degree of its variables, in general $\frac{m}{n} = p_1 \frac{n-1}{2}$.
- p is the probability of change used in DSA. DBA does not use this variable.

We can build on the definition for DCSPs to define DynDCSPs as follows [6]:

- We introduce a dynamics function: $\Delta : P_t \mapsto P_{t+1}$
- This expression produces a *sequence* of individual DCSP problems: $\mathbf{P} = \{P_0, P_1, \cdots, P_\infty\}$
- Given this, we can characterize the *rate* of change of problems as $\frac{dP}{dt} = \lim_{\Delta t \rightarrow 0} \frac{P_{t+\Delta t} - P_t}{\Delta t} = \frac{1}{\Delta t} \sum_{i=t}^{t+\Delta t} (\mid f_i^a \mid + \mid f_i^r \mid)$

The reader should note that the Δ function is associated with time and that it works by adding a set of constraints, f_i^a, and removing another set of constraints, f_i^r at each step. This means that we cannot always solve any given problem before it changes, so our goal becomes to maximize the number of satisfied constraints, rather than necessarily to find a complete solution or determine its impossibility. Also, if these sets become imbalanced (i.e. $|f_i^a| \neq |f_i^r|$) or the tightness of the constraints within these sets change, then the the problem can change in difficulty as well. We refer to these types of changes as second-order dynamics and do not consider them in this work, although it is an important topic for future investigation.

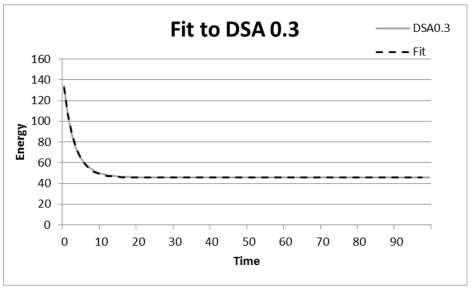


Figure 1: Function fit for DSA 0.3 (n=200, $|D_i|$=3, p_1=0.027, p_2=0.33)

Finally, we map DynDCSPs to thermodynamic theory.

- Variables are equivalent to particles.
- Constraints are equivalent to the interactions between particles.
- The quality of a solution—measured by the number of unsatisfied constraints—is equivalent to the energy of the system E.
- The entropy of a solution—measured by the number of assignments with a specific quality—is equivalent to the entropy of a physical system, generally measured by the number of states with a specific energy.

Mailler and Zheng found that this mapping allowed for a very close match between the empirical behavior of DynDCSP systems and that predicted by the thermodynamics-style equations implied by the mapping. First, they showed that the change in energy in a DynDCSP could be described effectively by Newton's cooling law [7].

$$E_Q = mp_2 - be^{-\frac{rate*t}{m}}$$

Here, b is a constant factor, t is time, and $rate$ is the rate of constraint-change at time t. By allowing A to represent the convergence rate, and B the convergence point, they then showed that the performance profile for distributed protocols could also be fit by the same function.

$$E_W = B + ke^{-\frac{t}{A}}, 0 < B < mp_2, 0 < k$$

This yields an expected change in energy per unit time of

$$\frac{dE_W}{dt} = \frac{B - E}{A}$$

Finally, using the First Law of Thermodynamics ($\Delta E = \delta Q - \delta W$) they showed that the expected equilibrium for a dynamically changing constraint system being simultaneously solved by a protocol is

$$E_0 = \frac{Amp_2 + \frac{mB}{rate}}{A + \frac{m}{rate}}$$

In practice, obtaining the values of A and B involves running numerous trials using static instances for various values of n, p_1, and p_2. The data produced by these runs is then processed. Processing involves computing the average energy for each time step over the test series, then calculating A and B using a program like Mathematica [10] to obtain a least squares fit. Figure 1 shows the result of this procedure for the DSA protocol with $p = 0.3, n = 200, |D_i| = 3$ for all $i, p_1 = 0.027, p_2 = 0.33$, which is fit quite well by $A = 2.95$ and $B = 45.7$.

3. ALGORITHM ANALYSIS

Being able to predict a system's dynamic performance using two parameters that are obtained by testing static instances is, in itself, a significant advance. However, we are still left with all of the parameters for static DCSPs— n, each $|D_i|$, p_1, and p_2— when evaluating a new protocol. Worse yet, some protocols like DSA have tunable parameters whose settings can be very problem specific. One way to address this problem is to perform exhaustive testing on the protocol in question using parameter settings that may be encountered in practice. The approach we follow here is to analyze the protocol to determine if estimates or bounds on its performance can be derived.

In this section, we present an analysis of two popular distributed protocols; DSA and DBA. For both of these protocols, we will derive functions that describe their convergence rate A as a function of problem density and tightness. Because the DSA and DBA algorithms remain fundamentally unchanged in the transition to their dynamic counterparts, our analysis, like the empirical studies described above, can provide valuable insights into their dynamic performance, even though they don't address the dynamic case specifically.

Note also that we necessarily address only the average-case behavior for these algorithms. For any DCSP algorithm, it is always possible to produce problem instances that converge arbitrarily fast, or not at all. As such, absolute bounds on algorithm performance are generally unhelpful for us, and we focus instead on characterizing how in the large scale each algorithm solves problems on average.

3.1 Distributed Stochastic Algorithm

The Distributed Stochastic Algorithm (DSA) is perhaps the simplest of all of DynDCSP algorithms (see figure 2). Because of its simplicity, it can remain fundamentally unchanged by the transition from a DCSP setting to that of DynDCSP, so it accommodates much simpler analysis approaches. The protocol works by having each agent determine if they could reduce the number of constraint violations by changing their variable's value. If they can, with some probability p, they change their value and inform their neighbors. There are several variants to DSA, with the most popular being DSA-B. In addition to operating like DSA, DSA-B also has a possibility of making lateral moves (i.e., changes when the potential for improvement is 0) if its constraints are not satisfied.

The optimal setting for p has been the subject of some experimentation [12, 5], and is generally believed to lie somewhere around 0.3. Prior studies, however, focused exclusively on the "final" quality produced by the protocol on static instances, where a slower approach to obtaining a solution might yield a superior answer, rather than on the convergence rate.

We expect the value of p to be strongly associated with the convergence rate of the algorithm, since if the value of p is small enough, agents will only change infrequently and converge on a final result after a longer time. However, if p is too large, DSA agents can cancel out each other's intended improvements—if both agents of a constraint change their values, it may or may not actually satisfy the constraint or cause an improvement. With DSA-B, we have to take into account the effect of a lateral move, which is more subtle. Lateral moves are not expected to directly affect the convergence rate, but can potentially affect the convergence point by allowing the algorithm to explore different regions of the search space.

```
procedure main
  while (not terminated) do
    update agent_view with incoming
      ok? (x_j, d_j) messages;
    new_value ← choose_value;
    if new_value ≠ d_i do
      d_i ← new_value;
      send ((ok?, (x_i, d_i)) to all x_j ∈ neighbors;
    end if;
  end do;
end main;

procedure choose_value
  if d_i has no conflicts do
    return d_i;
  v ← the value with the least conflict (v ≠ d_i);
  if v has fewer conflicts than d_i
     and random < p do
    return v;
  else
    return d_i;
end choose_value;
```

Figure 2: The procedures of the DSA algorithm.

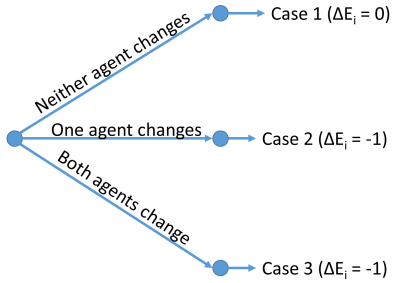


Figure 3: Probability tree for the upper bound analysis.

Our basic methodology in analyzing DSA is constraint-centric, and takes a whole-system perspective, in which the number of unsatisfied constraints is always decreasing. Imagine a single constraint within the problem. At any given moment, this constraint is either satisfied or unsatisfied. If it is satisfied, then we do not consider it further. For other binary constraints, one of three things can occur (see figure 3); both agents do nothing, only one of the agents changes, or both agents change (often called a collision). Because variable changes are determined by the setting of p, it allows us to state simply that, taking the system as a whole, the probability for each of the cases for each unsatisfied constraint is $(1-p)^2$, $2p(1-p)$, and p^2, respectively. However, the resulting energy change of making these moves is dependent on the circumstances of each case.

Note that the convergence point B for a protocol represents the

final energy that it can reach, given the current system parameters, on average. Therefore, we can assume that when the system has not yet reached the convergence point, there will on average always be an agent that can make an improvement, but not beyond that point. As such, we here make an overriding assumption that applies throughout our analysis— namely, at each time step, one or more of the agents *will* attempt to change values, as long as the system energy has not reached the convergence point. Without this assumption, the analysis would assume that every problem is eventually solvable, resulting in a sizable overestimate of the energy reduction per unit time.

Upper Bound Analysis.

We continue our analysis of DSA by attempting to derive an upper (optimistic) bound on how fast it can reduce the energy of a problem over time. To create our upper bound approximation, we have to employ another simplifying assumption— that when an agent changes value, it always leads to a reduction in energy. This first assumption is not unreasonable, since DSA only changes values if the energy can be reduced, but it ignores the fact that an agent may change its value to solve a different constraint then the one being considered. It also ignores the collision case, where two agents change their values at the same time and the constraint remains unsatisfied. Because these two caveats slow energy reduction, however, this is suitable for an upper bound. The resulting energy-change values for each case are represented in figure 3.

With these assumptions, we can compute an upper bound on the average convergence rate for DSA:

$$\frac{dE_w}{dt} \leq (E-B)*[(0)*(1-p)^2+(-1)*(2p(1-p))+(-1)*p^2]$$

$$\frac{dE_w}{dt} \leq (E-B)*[-2p+2p^2-p^2]$$

$$\frac{dE_w}{dt} \leq (E-B)*[p^2-2p]$$

Lower Bound Analysis.

The lower bound analysis is similar to that for the upper bound, except that it changes certain assumptions (see figure 4). First, when just one agent alters its value, we consider that the structure of the constraints will not always allow the removal of a violation. Instead, the expected likelihood of solving a constraint is $1-p_2$, in which case the energy decreases by one, producing an expected outcome of $\Delta E_i = (-1)*(1-p_2) = -1+p_2$. Second, we assume that when two agents change their values at the same time, the constraint between them is never satisfied. Both of these are quite pessimistic estimates of the actual behavior.

These assumptions lead to the lower bound on convergence rate for DSA:

$$\frac{dE_w}{dt} \geq (E-B)*[(0)*(1-p)^2+(-1+p_2)*(2p(1-p))+(0)*p^2]$$

$$\frac{dE_w}{dt} \geq (E-B)*[(p_2-1)(2p-2p^2)]$$

Approximate Analysis.

Although it is instructive to have upper and lower bounds, it is also much nicer to have an estimate of the algorithm's performance in practice. One way to achieve this is to assume that the actual

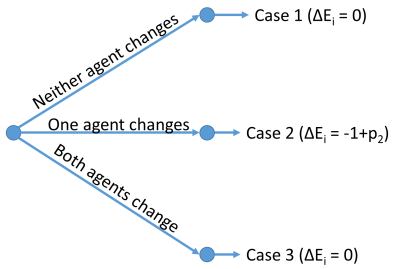


Figure 4: Probability tree for the lower bound analysis.

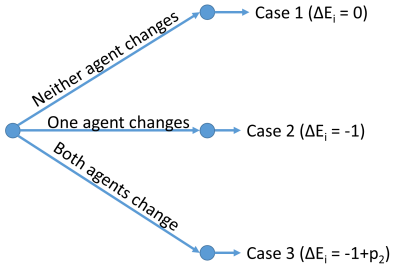


Figure 5: Probability tree for approximate analysis.

convergence rate lies at the average of the predicated upper and lower bounds. For many circumstances, this is entirely reasonable. However, we can do better through a careful choice of terms. In considering the probability tree for our lower bound analysis, one might note that if one agent moves, it is actually more likely than just $1-p_2$ to solve the constraint in question, while if two agents move at the same time, they are at least somewhat likely to inadvertently solve the constraint. For our improved approximation, we assume that the constraint is solved (leading to a reduction in energy of 1) whenever only one agent changes value, and that if both agents change values, there is a $1-p_2$ probability that the constraint will be satisfied (see figure 5).

Having made these assumptions, we can create a simple approximation of energy change which follows directly from the probability tree:

$$\frac{dE_w}{dt} \approx (E-B)*[(0)*(1-p)^2+(-1)*(2p(1-p))+(-1+p_2)*p^2]$$

$$\frac{dE_w}{dt} \approx (E-B)*[-2p+2p^2-p^2+p_2p^2]$$

$$\frac{dE_w}{dt} \approx (E-B)*[(1+p_2)p^2-2p]$$

Experimental Results.

To evaluate our bounds and approximation, we conducted a series of experiments on static, DCSP instances using DSA. For our test setup, we chose to use $n = 200$ variables, with the domain of each variable v_i containing $|D_i| = 3$ values. We chose a fixed value of $p_2 = 0.33$ and varied the density of the problems $p_1 = \{0.02, 0.023, 0.027\}$. We tested DSA with probability settings in $p = \{0.3, 0.5, 0.7\}$, and for each combination of p_1 and p tested 100 random instances, measuring the number of constraint violations at each step. This resulted in 900 test runs overall.

```
when received (ok?, x_j, d_j) do
   add (x_j, d_j) to agent_view;
   when received ok? from all neighbors do
      send_improve;
      mode ← wait_improve;
   end do;
end do;

procedure send_improve
   compute best assignment and improve value;
   new_value ← best assignment;
   my_improve ← improve;
   send improve to neighbors;
end send_improve;
```

Figure 7: The procedures of the *wait_ok?* mode in Distributed Breakout.

Once we collected the data, we analyzed it using the method described at the end of section 2. The result of this analysis is a set of values for A and B, which can be seen in table 1. The values in this table show two general trends. First, as the problems become more dense, the problem solvers produce worse solutions, and do so at a slower pace. Both of these are to be expected, since denser problems decrease the likelihood of a variable being improvable at each step, and result in more constraints that cannot be satisfied. Interestingly, though, as the value of p increases, the convergence rate decreases (recall that smaller rate values mean faster convergence). Thus, despite the associated increase in the number of collisions that occur, the DSA with $p = 0.7$ operates faster than DSA with $p = 0.3$ and is therefore expected to perform better in highly dynamic setting.

	Density					
	0.201		0.023		0.027	
	A	**B**	**A**	**B**	**A**	**B**
DSA 0.3	2.47	24.89	2.70	32.96	2.94	45.81
DSA 0.5	1.51	24.92	1.58	32.17	1.72	44.37
DSA 0.7	1.14	23.00	1.20	30.90	1.28	42.17

Table 1: Measured values for convergence rate and point for DSA.

Using these values of A and B, we next plotted the convergence rate function along with our lower bound, upper bound, and approximation functions (see figure 6). Because of space limitations, we only show the graphs for density $p_1 = 0.027$, but the results for other configurations are similar.

These graphs show a plot of the expected energy change as a function of the current energy. In all of these graphs, the lower and upper bounds bracket the function fit as expected. The estimates that these bounds provide naturally diverge from the actual behavior considerably as the energy of the problem increases, since they are overly pessimistic and optimistic, respectively. The approximation function does fairly well at predicting the behavior of the protocol. For low values of p, the function tends to overestimate the energy reduction that is obtained, likely because it underestimates the impact of collisions, and overestimates the impact for single variable changes (again, it assumes that the change will fix the violation). As the value of p increases, the estimate becomes more accurate, and by the time the value of $p = 0.7$, we see that it is slightly underestimating the true value.

3.2 Distributed Breakout Algorithm

The Distributed Breakout Algorithm (DBA) [11] is a distributed adaptation of the centralized Breakout algorithm [9]. DBA works by alternating between two modes. The first mode (see figure 7) is called the *wait_ok?* mode. During this mode, agents collect *ok?* messages from each of their neighbors containing their current variable assignments. The agents then calculate the best new value for their own variables, along with the improvement to their local evaluation. The agents then send out *improve?* messages containing this improvement value to each of their neighbors and change to the *wait_improve?* mode.

```
when received (improve, x_j, improve, eval) do
   record message;
   when received improve from all neighbors do
      send_ok;
      clear agent_view;
      mode ← wait_ok;
   end do;
end do;

procedure send_ok
   if my_improve is better than all of my neighbors
      current_value ← new_value;
   end if;
   when in a quasi-local-minimum do
      increase the weights on all violated constraints;
   end do;
   send (ok?, x_i, current_value) to neighbors;
end send_ok;
```

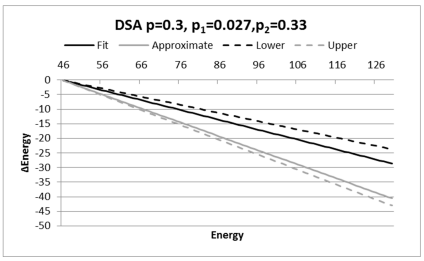
Figure 8: The procedures of the *wait_improve?* mode in Distributed Breakout.

In the *wait_improve* mode (see figure 8), the agents collect *improve?* messages from each of their neighbors. When all of these messages are received, the agents check to see if they have the best improvement amongst their neighbors. If an agent does, it is allowed to change its value. Finally, the agents send *ok?* messages to each of their neighbors and change back to the *wait_ok?* mode. The algorithm starts up with each agent sending *ok?* messages and going into the *wait_ok?* mode. By repeating this protocol again and again, the agents hill-climb to a better state.

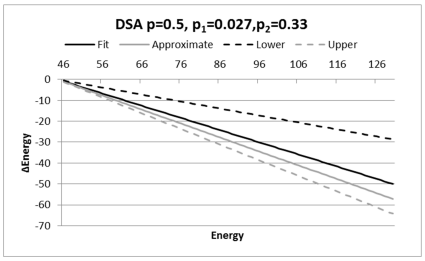
However, like all hill-climbing algorithms, DBA is prone to getting trapped in local minima, so it employs a rather unusual escape mechanism. If an agent has at least one violated constraint, is not able to change its value to remove the violations, and has no neighbor that reports it can improve, then it is in a state called a quasi-local-minimum (QLM). A QLM is a weaker condition than what is typically thought of as a local minimum, since QLM *can* occur when the search is not in a true local minima. However, if the search is in a true local minima, it is guaranteed to be detected by the QLM mechanism.

When an agent believes it is in a quasi-local-minimum (QLM), it increases the weights on each of its violated constraints, placing higher priority on solving constraints with a higher weight. The DBA protocol without the QLM escape mechanism is sometimes referred to as the Maximum Gain Message (MGM) protocol [2].

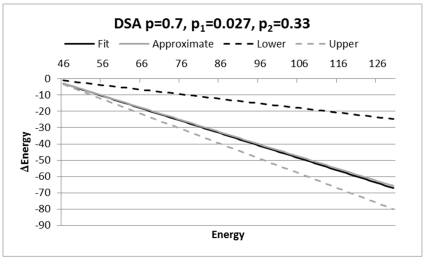
Because of the strict locking mechanism employed in the algorithm, the overall behavior of the agents is to simultaneously switch back and forth between the two modes. If one or more of the agents reacts slowly or messages are delayed, the neighboring agents wait for the correct message to arrive. This also makes the protocol's communication usage very predictable, since in each mode, each agent sends exactly one message to each of its neighbors. Thus, if there are m constraints, exactly $2m$ messages are transmitted during each step.



(a) DSA 0.3



(b) DSA 0.5



(c) DSA 0.7

Figure 6: Comparison of function fit versus lower, upper, and approximate functions for DSA

Protocol Analysis.

DBA is a more complex algorithm than DSA, but its analysis actually proves to be much simpler. We began our analysis by observing that because of the strict locking mechanism, the average degree of the problem is the limiting factor on the convergence rate. Taking a variable-centric view of the analysis, we started by estimating the maximum number of agents that can move during a single time step. Since only one out of a set of neighbors can move, we have:

$$moves \leq \frac{n}{degree} = \frac{n}{\frac{m}{n}} = \frac{n^2}{m}$$

Since we know how many variables are going to change, we need only estimate the impact of those changes. And since this is an upper bound approximation, we can assume that if a variable changes its value, all of the currently unsatisfied constraints that could be satisfied will be. This is not a particularly bad estimate, because we know that DBA will choose to change the variables that lead to the greatest overall reduction in energy.

To calculate the energy decrease for a single variable change, we only need to know how many constraints the variable has (which is simply the average degree), and the probability that each of those constraints is currently violated (i.e., the average energy of each constraint, which is the current energy divided by the number of constraints). However, since we know that we have to account for any unsatisfiable constraints in the problem, we subtract the convergence point from the current energy level.

What this leads to is that change in energy is the number of variable changes times the number of constraints per variable times the probability that those constraints can become satisfied. The formula is therefore:

$$\frac{dE_w}{dt} \leq \frac{n^2}{m} * \frac{m}{n} * \frac{E-B}{m} = \frac{n(E-B)}{m}$$

This formula is remarkably satisfying because it can easily be rewritten as $\frac{dE_w}{dt} \leq \frac{E-B}{degree}$, which says that the convergence rate is directly limited by the average degree of the problem, as we expect.

Experimental Results.

In a similar way to how we tested our bounds and approximation of DSA, we conducted a series of experiments on DBA. As before, we used $n = 200$ variables each with $|D_i| = 3$ values. We used a fixed value of $p_2 = 0.33$, and varied the density of the problems $p_1 = \{0.02, 0.023, 0.027\}$. For each of the values of p_1, we generated 100 random instances, and counted the number of constraint violations at each step. This test series consisted of 300 runs.

Following our standard procedure, we fit the results to Newton's cooling equation, and derived values for both A and B. These values are presented in table 2. Like DSA, DBA is affected by the density of the problem. However, density has a greater impact on the convergence rate of DBA than it does on DSA. This can be easily understood when we consider that density and degree are closely related, and that the higher the degree, the fewer the number of variables that can change during a time step.

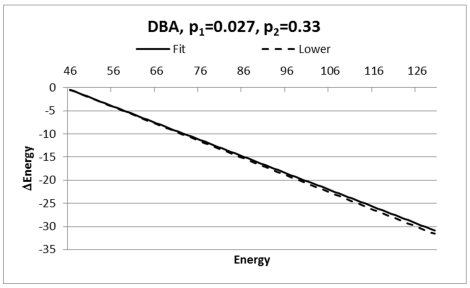


Figure 9: Comparison of function fit versus lower bound for DBA

	Density					
	0.201		0.023		0.027	
	A	**B**	**A**	**B**	**A**	**B**
DBA	1.97	24.92	2.37	33.54	2.77	44.72

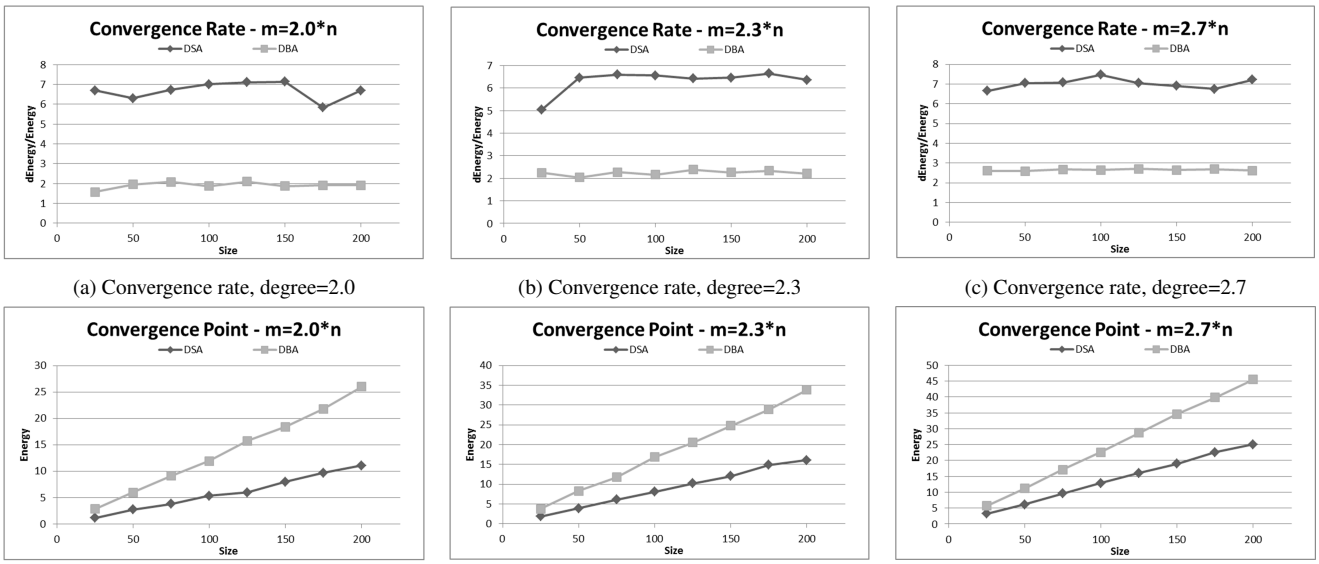
Table 2: Measured values for convergence rate and point for DBA.

In figure 9, the results of the comparison between the fit and the upper bounds is shown. The graph clearly shows a very tight bounds on the actual performance of the protocol. Because our upper-bound analysis provides such a tight match, we decided not to pursue creating lower bounds or a tighter approximation.

3.3 Effects of Problem Size

As we mentioned in the introduction of this paper, our motivation for this work is to reduce the amount of empirical evaluation needed to estimate a protocol's performance. The work of Mailler and Zheng made it possible to eliminate one parameter by creating a procedure for estimating performance in dynamic environments. Our analysis thus far has shown that we can estimate the value for the parameter A given the value of p, for DSA, or the average degree for DBA. What's interesting about that statement is that our equations for convergence rate are all scale-independent. This is an important finding, because it says that if we determine the rate for a problem of one size, then as long as we don't change the degree or tightness of the problem as we scale it, the convergence rate A should remain unchanged.

To verify this interpretation, we ran another series of tests, varying the size of the problem from $n = 25 - 200$, for three set-



(a) Convergence rate, degree=2.0

(b) Convergence rate, degree=2.3

(c) Convergence rate, degree=2.7

(d) Convergence point, degree=2.0

(e) Convergence point, degree=2.3

(f) Convergence point, degree=2.7

Figure 10: Convergence rate and point versus problem size, for three problem degrees.

tings of average degree $\frac{m}{n} = \{2.0, 2.3, 2.7\}$ and a fixed tightness $p_2 = 0.33$. As in our previous experiments, we generated 100 random instances, produced performance profiles, and fit those profiles to Newton's cooling function. The results can be seen in figure 10.

As we suspected, the convergence rate of the protocols is unaffected by scale. Intuitively, this makes perfect sense, because both of these protocols work in a parallel manner. If we assume that on average the energy of the system is evenly spread, then it allows the agents to work nearly independently of one another.

When looking at the convergence point, another trend becomes apparent— the value of B scales linearly with the size of the problem. To understand why this occurs, consider two independent DCSPs, P_1 and P_2, that have the same values for n, each $|D_i|$, p_1, and p_2, but not necessarily the same structure. Now since P_1 and P_2 are statistically the same, if we ran DSA on them both, we would expect that the convergence points of the final solutions DSA produces, B_1 and B_2, would be, on average, the same.

Now, it is well understood that if you combine two independent thermodynamic systems together, the energy of the combined system is the sum of the energies of the individual systems, due to the law of conservation of energy [4]. In a similar manner, if we combine P_1 with P_2 without changing the ratio of constraints to variables, then the convergence point would become $B_1 + B_2 = 2 * B_1$.

For us, the implication is simple. If we take a static DCSP instance with a known degree, tightness, and domain size and compute the value for B, then we can extrapolate the value for B for any size problem. Combining this with our analysis, we can then recreate the performance profile, and predict the protocols' behavior in dynamic environments, without needing to perform additional empirical testing.

4. CONCLUSION AND DISCUSSIONS

In this paper, we made use of a thermodynamic analysis of DynDCSPs to develop analytical predictions—both approximations and bounds—for the behavior of DSA and DBA, two well-known algorithms for the problem. Through experimentation, we showed that these analyses are able to predict the algorithms' behavior with a high degree of accuracy. Having the ability to generate close predictions of convergence rate and convergence point is a huge asset to the development of DCSP systems, since it allows us to form a reasonable expectation of system behavior in advance of actual testing. Insight into the algorithms' convergence rate is also especially important, since it allows us to understand how they will behave in dynamic systems.

We also found that convergence point and convergence rate parameters, generated by testing with a single problem instance, can be reliably predicted for other problem sizes—convergence point increases linearly, while convergence rate generally remains constant. This is important, since it means that we are now able to derive parameters from testing with a single problem instance that allow us to accurately extrapolate algorithm behaviors for any size problem with the same density and parameter values.

In future work, we'd like to develop similar analyses for other well-known DCSP algorithms. This type of analysis is naturally deeply dependent on finding useful approaches based on each algorithm's structure, but we are confident that our current work provides a strong foothold to help approach this later research. We also plan to address the second-order dynamics of DCSPs—the effects of changes in density and tightness on algorithm behavior. This analysis will aid us in developing a fuller understanding of DCSP algorithms, and afford us power to extrapolate over a much wider variety of problem instances. Additionally, we'd like to look at a more formal development of hard bounds on these algorithms, as well as ways that we could tighten our overall approximations.

5. ACKNOWLEDGMENTS

We are deeply appreciative of the help of the National Science Foundation and the Air Force Research Laboratory for funding our work on this research.

This material is based on research sponsored by the Air Force Research Laboratory, under agreement number FA8750-13-1-0124 and the National Science Foundation under Grant No. IIS-1350671. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily

representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government.

REFERENCES

[1] F. Bacchus and P. van Beek. On the conversion between non-binary constraint satisfaction problems. *Proceedings of the 15th Nat'l/10th Conf. on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, pages 311–318, 1998.

[2] E. Bowring, J. P. Pearce, C. Portway, M. Jain, and M. Tambe. On k-optimal distributed constraint optimization algorithms: New bounds and algorithms. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2*, pages 607–614, 2008.

[3] R. Dechter and A. Dechter. Belief maintenance in dynamic constraint networks. In *Proceedings of the 6th National Conference on Artificial Intelligence (AAAI-88)*, pages 37–42, 1988.

[4] R. Fitzpatrick. *Thermodynamics and Statistical Mechanics: An intermediate level course*. Lulu Enterprises, Inc., 2006.

[5] S. Fitzpatrick and L. Meertens. *Distributed Sensor Networks: A Multiagent Perspective*, chapter Distributed Coordination Through Anarchic Optimization, pages 257–294. Kluwer Academic Publishers, 2003.

[6] R. Mailler. Comparing two approaches to dynamic, distributed constraint satisfaction. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 1049–1056, 2005.

[7] R. Mailler and H. Zheng. A new analysis method for dynamic, distributed constraint satisfaction. In *Proceedings of the 2014 International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 901–908, 2014.

[8] A. Meisels, I. Razgon, E. Kaplansky, and R. Zivan. Comparing performance of distributed constraints processing algorithms. In *Proc. AAMAS-2002 Workshop on Distributed Constraint Reasoning DCR*, pages 86–93, 2002.

[9] P. Morris. The breakout method for escaping local minima. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 40–45, 1993.

[10] I. Wolfram Research. Mathematica version 8.0, 2010.

[11] M. Yokoo and K. Hirayama. Distributed breakout algorithm for solving distributed constraint satisfaction problems. In *Proceedings of the 2nd Int'l Conf. on Multiagent Systems*, pages 401–408, 1996.

[12] W. Zhang, G. Wang, Z. Xing, and L. Wittenburg. Chapter 13: A comparative study of distributed constraint algorithms. *Distributed Sensor Networks: A Multiagent Perspective*, 2003.